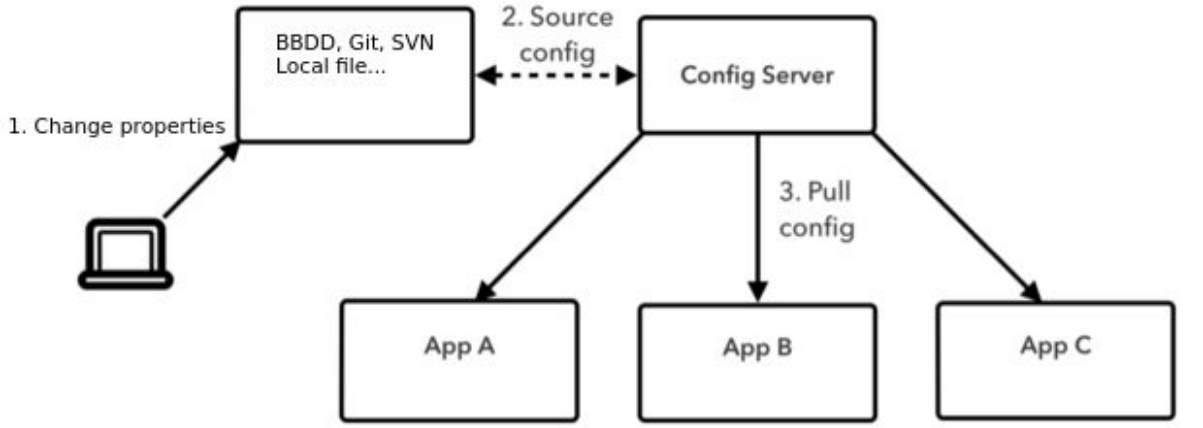
Actualización properties en caliente

Este esquema muestra lo que se quiere conseguir:



1. Realizamos un cambio en el fichero propiedades y hacemos un push al repositorio Git.
2. El Config Server ve ese cambio y actualiza el “Environment” de Spring.
3. Esos cambios se propagan a las aplicaciones, refrescando las propiedades y los beans sin necesidad de reiniciar servidores.

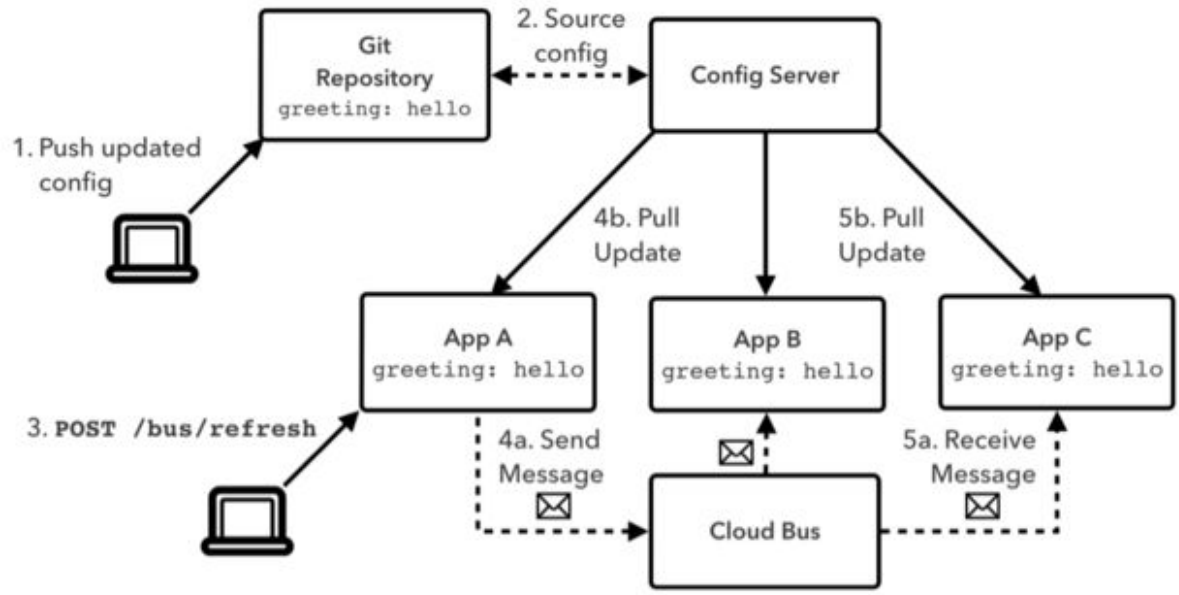
Solución: Spring Cloud Config y Spring Cloud Bus

Spring Cloud Config es un proyecto de Spring que nos permite externalizar la configuración en sistemas distribuidos. Nos facilita la creación de un Config Server para el control de actualizaciones y el refresco del contexto en las aplicaciones clientes.

Spring Cloud Bus es un proyecto de Spring que nos permite unir nodos de un sistema distribuido a una cola de mensajes.

Existen dos posibilidades para el refresco de propiedades y beans. Si no existe ningún bean con anotación **@RefreshScope**, Spring Cloud llamará a refreshAll(), actualizando todo el contexto. La otra opción es añadir la anotación a los beans que sean propensos a actualizarse, de esta forma, solo refrescamos los beans seleccionados.

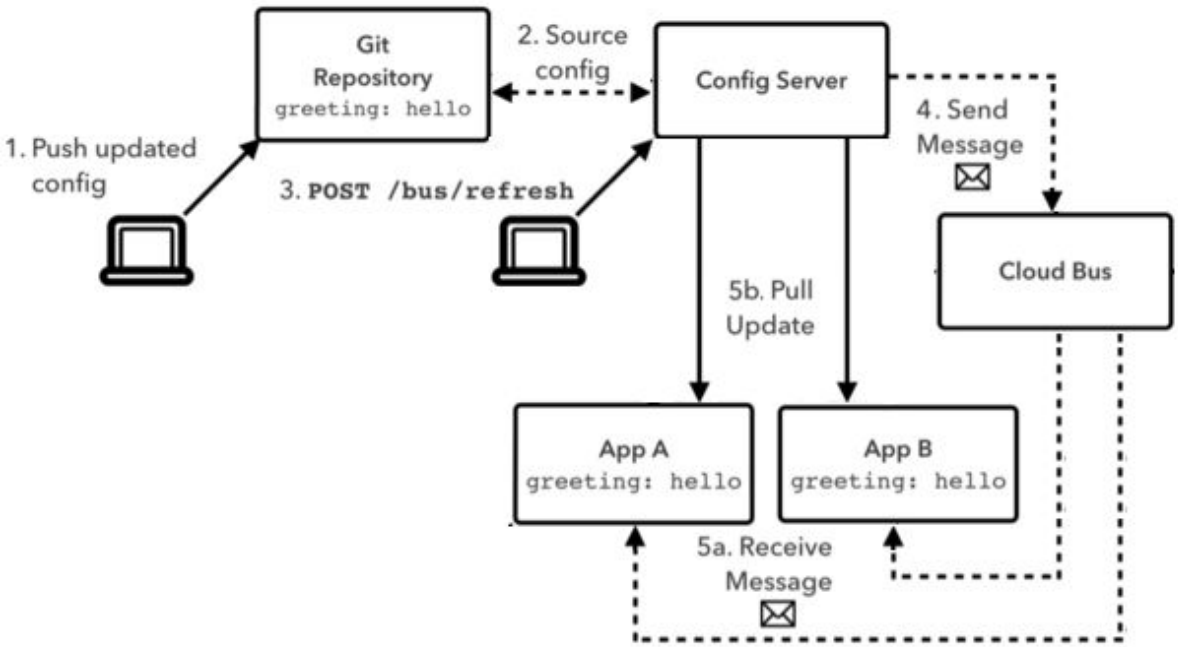
En el siguiente esquema se muestra la solución aportada por Spring Cloud Config y Spring Cloud Bus:



---

1. Realizamos los cambios en el fichero de propiedades y realizamos push al repositorio Git.
2. El Config Server (Spring Cloud Config Server) recibe estos cambios y actualiza el contexto de Spring. Estos cambios no se propagan automáticamente a las aplicaciones cliente, de modo que nos permite tener un control más estricto de cuándo queremos que empiecen a afectar los cambios realizados.
3. Para propagar los cambios, debemos llamar al endpoint */bus/refresh*. En este esquema llamamos al endpoint disponible en cualquier nodo.
4. La llamada a este endpoint inicia el trabajo con el Cloud Bus (Spring Cloud Bus):
    a. Se envía un mensaje al Bus indicando que hay actualizaciones pendientes.
    b. El nodo al que hemos llamado descarga y actualiza el entorno desde el Config Server.
5. El resto de nodos inicia también el trabajo con el Cloud Bus:
    a. recibir el mensaje de que hay actualizaciones pendientes.
    b. descargar y actualizar el entorno desde el Config Server.

En el siguiente esquema se muestra nuestra solución:



1. Realizamos los cambios en el fichero de propiedades y realizamos push al repositorio Git.
2. El Config Server (Spring Cloud Config Server) recibe estos cambios y actualiza el contexto de Spring. Estos cambios no se propagan automáticamente a las aplicaciones cliente, de modo que nos permite tener un control más estricto de cuándo queremos que empiecen a afectar los cambios realizados.
3. Llamamos al endpoint */bus/refresh* que tenemos disponible en el Config Server.
4. En este caso, es el Config Server el que envía mensaje de actualización disponible al Cloud Bus (Spring Cloud Bus).
5. Una vez se envía el mensaje al Bus, los nodos:
    a. reciben el mensaje de actualización disponible.
    b. descargan la actualización y refrescan el entorno.

PoC

Se han creado tres proyectos en Git:

- http://54.76.206.84/magicbox/config-server -> Proyecto con el Config Server
- http://54.76.206.84/magicbox/config-repo -> Repositorio que contienen los ficheros de propiedades.
- http://54.76.206.84/magicbox/config-client -> Proyecto cliente para las pruebas de Config Server.

Para trabajar con Cloud Bus se ha creado una instancia en CloudAMQP para tener una cola RabbitMQ para el PoC. Tienen disponible un RabbitMQ Management para comprobar conexiones y mensajes y utilizan servicios de AWS para las instancias. Es una capa gratuita, que limita el uso a 20 conexiones simultáneas y un máximo de un millón de mensajes mensuales. Los datos de la instancia son:

- Host: hare.rmq.cloudamqp.com
- Vhost: vhqxzlpi
- Username: vhqxzlpi
- Password: Hf0w3k2XnDo1hzt9EING_Esw7t5-fZx-

En el repositorio de los archivos de configuración, la nomenclatura de los ficheros de configuración siguen el estándar de Spring:

Nomenclatura	**Ejemplo**
application.properties application.yml application-profile.properties application-profile.yml	myApp.properties myApp.yml myApp-development.properties myApp-development.yml

---

Consideraciones

- ¿Qué ocurre con el refresco mientras el bean está en un proceso?

- Revisar el proceso que sigue Spring Cloud Config Client para refrescar los beans.

- Probar un cliente sin Spring Boot, para ver si la configuración necesaria cambia mucho, ya que Spring Boot tiene mucha configuración predefinida.